BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

WHAT IS CLAIMED IS:

1. A processor comprising:

an execution unit to execute instructions;

a replay system coupled to the execution unit to replay instructions which have not executed properly, the replay system comprising:

a checker to determine whether each instruction has executed properly; and

a plurality of replay queues, each replay queue coupled to the checker to temporarily store one or more instructions for replay.

2. The processor of claim 1 wherein said plurality of replay queues comprises:

a first replay queue coupled to the checker to temporarily store instructions corresponding to a first thread; and

a second replay queue coupled to the checker to temporarily store instructions corresponding to a second thread.

3. A processor comprising:

an execution unit to execute instructions;

a replay system coupled to the execution unit to replay instructions which have not executed properly, the replay system comprising:

a checker to determine whether each instruction has executed properly; and

a replay queue coupled to the checker to temporarily store one or more instructions of a plurality of threads for replay, the replay queue partitioned into a plurality of sections, each section provided for storing instructions of a corresponding thread.

4. The processor of claim 3 wherein said plurality of replay queue sections comprises:

a first replay queue section coupled to the checker to temporarily store instructions corresponding to a first thread; and

a second replay queue section coupled to the checker to temporarily store instructions corresponding to a second thread.

5. The processor of claim 3 wherein said replay system further comprises:

a replay loop to route an instruction which executed improperly to an execution unit for replay; and

a replay queue loading controller to determine whether to load an improperly executed instruction to the replay loop or into one of the replay queue sections.

6. The processor of claim 3 and further comprising:

a scheduler to output instructions; and

a multiplexer or selection mechanism having a first input coupled to the scheduler, a second input coupled to the replay loop and a plurality of additional inputs, each additional input coupled to an output of one of the replay queue sections.

7. The processor of claim 3 wherein each said replay queue section comprises a replay queue section coupled to the checker to temporarily store one or more long latency instructions of a thread until the long latency instruction is ready for execution.

8. The processor of claim 3 wherein each replay queue section comprises a thread-specific replay queue section coupled to the checker to temporarily store an instruction in which source data must be retrieved from an external memory device, the instruction being unloaded from the replay queue section when the source data for the instruction returns from the external memory device.

9. The processor of claim 3 wherein said execution unit is a memory load unit, the processor further comprising:

a first level cache system coupled to the memory load unit;

a second level cache system coupled to the first level cache system; and

wherein the memory load unit performs a data request to external memory if there is a miss on both the first level and second level cache systems.

10. The processor of claim 9 wherein a load instruction of a thread will be loaded into a replay queue section corresponding to the thread when there is a miss on both the first level and second level cache systems for the load instruction, and the load instruction is unloaded from the

replay queue section corresponding to the thread for re-execution when the data for the instruction returns from the external memory.

11. A processor comprising:

a multiplexer having an output;

a scheduler coupled to a first input of the multiplexer;

an execution unit coupled to an output of the multiplexer;

a checker coupled to the output of the multiplexer to determine whether an instruction has executed properly;

a plurality of thread-specific replay queue sections to temporarily store instructions for each of a plurality of threads, an output of each of the replay queue sections coupled to additional inputs of the multiplexer; and

a controller coupled to the checker to determine when to load an instruction into one of the replay queue sections and to determine when to unload the replay queue sections.

12. The processor of claim 11 and further comprising a staging section coupled between the checker and a further input to the multiplexer to provide a replay loop, the controller controlling the multiplexer to select either the output of the scheduler, the replay loop or an output of one of the replay queue sections.

13. The processor of claim 11 wherein the controller determines when to unload one or more of the replay queue sections based on a data return signal.

14. A method of processing instructions comprising:

dispatching an instruction where the instruction to an execution unit and to a replay system;

determining whether the instruction executed properly;

if the instruction did not execute properly, then:

determining whether the instruction should be routed back for re-execution or whether the instruction should be temporarily stored based on a thread of the instruction.

15. A method of processing instructions comprising:

dispatching an instruction where the instruction is received by an execution unit and a replay system;

determining whether the instruction executed properly;

if the instruction did not execute properly, then:

routing the instruction to the execution unit for re-execution if the instruction is a first type of instruction;

otherwise, loading the instruction into one of a plurality of thread-specific replay queue sections based on a thread of the instruction if the instruction is a second type of instruction.

16. The method of claim 15 wherein the first type of instruction comprises a short latency instruction, and the second type of instruction is a longer latency instruction.

17. A method of processing instructions comprising:

initially allocating execution resources for multiple threads;

determining that a first thread has stalled;

temporarily storing one or more instructions of the first thread in a queue; and

continuing to allocate execution resources to other threads which have not stalled

18. The method of claim 17 wherein said step of continuing to allocate comprises the step of continuing to allocate execution resources to other threads which have not stalled and inhibiting the allocation of further resources to the stalled thread by temporarily storing the stalled thread instructions in the queue.

19. The method of claim 17 wherein priority for execution resources are allocated to the other threads which have not stalled on a rotating priority basis.

20. The method of claim 17 and further comprising the steps of:

detecting that the first thread is no longer stalled;

unloading the one or more instructions of the first thread from the queue; and

re-allocating at least some execution resources to the first thread.

21. The method of claim 17 wherein the step of determining that a first thread has stalled comprises detecting a long latency or agent instruction for the first thread.